WORKMAN NYDEGGER
A PROFESSIONAL CORPORATION
ATTORNEYS AT LAW
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UTAH 84111
TELEPHONE (801) 533-9800
FAX (801) 328-1707

TELECOPIER COVER SHEET

February 18, 2009

Total Number of Pages
(including cover letter): 12 pages

Please deliver the transmitted facsimile pages to:

Examiner Zheng Wei

United States Patent & Trademark Office

Business Phone	(571) 270-1059
Telecopier Phone:	(571) 270-2059
From:	Brian Tucker (Reg. No. 61,550)
Comments:	Draft Amendments to be discussed during in person interview on Tuesday, March 3rd at 1:30 EST
Serial No.	10/837,176
Docket No.	13768.783.74.1

****

We are transmitting from a Sharp FO-750 or Sharp FO-6100 facsimile machine. If you do not receive all the pages or they are unreadable, please contact me as soon as possible at (801) 533-9800.

BRIAN D. TUCKER
OFFICE DIRECT: 801-322-8427
EMAIL: btucker@WNLAW.com

VIA eFILE

PATENT APPLICATION
Docket No. 13768.783.74.1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of		)
	Michael E. Markley, et al.	)
Serial No.:	10/837,176	) Art Unit) 2192
Filed:	May 1, 2004	)
Conf. No.:	2171	)
For:	DETERMINING A MAXIMAL SET OF DEPENDENT SOFTWARE UPDATES VALID FOR INSTALLATION	)
Examiner:	Zheng Wei	)
Customer No.:	47973	)

AMENDMENT "F" AND RESPONSE
AFTER FINAL WITH RCE

VIA eFILE AMENDMENT
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In response to the FINAL Office Action of January 2, 2009 (paper no. 20081221), please amend the above-identified application as follows:

Amendments to the Claims are reflected in the listing of claims which begins on page 2 of this paper.

Remarks/Arguments begin on page 7 of this paper.

AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) In a computing system having ~~software~~an operating system installed, the operating system being divided into a plurality of discrete packages, and wherein a plurality of version updates are available to update a first package of the installed operating system~~software~~, a method for ~~updating software stored on a mobile computing device having an embedded operating system by~~ selecting an optimal set of available versions to allow for maximal version updates of ~~an installed~~the first package in the fewest update steps possible, while honoring package dependency constraints, the method comprising:

downloading, to a mobile computing device, a collection of update packages for updating an operating system that is installed on the mobile computing device, wherein the operating system is divided into a plurality of discrete packages that may each be updated independently from the other packages, and wherein the collection of update packages includes a plurality of update packages that correspond to different versions of a first package of the operating system that is already installed on the mobile computing device, and wherein each update package includes an associated manifest file that describes the contents and related dependency information of the update package, including the version of the update package;

~~reviewing a collection of update packages that are ready to be installed on a mobile computing device to update one or more currently installed packages, each update package having an associated manifest file that describes the contents and related dependency information of the update package, including the version of the update package;~~

for each package of the operating system currently installed on the mobile computing device, generating a graph having a node for the installed version of the package, the graph being generated from a manifest file corresponding to ~~each~~ the installed version of the package~~, the manifest file describing one or more other versions from which the version depends~~;

for each of the update packages to be installed, accessing a~~the associated~~ manifest file to determine on which installed package the update package depends, wherein each update package may

depend on an installed package by being directly dependent on the installed package or by being recursively dependent on another update package which in turn is dependent either directly on the installed package or on another update package ~~corresponding to the update package to determine that one of the generated graphs corresponds to the update package~~, and upon determining on which installed package the update package depends~~that the update package depends on a version of the package for which a node of the graph already exists~~, adding a new node to the graph, the new node corresponding to the version of the update package, such that upon adding a new node for each of the update packages, each graph is formed as a tree having the node for the installed version at the base of the graph and branches for each of the new nodes, each new node being linked to another node on which it depends, and each new node of the same graph corresponding to a different version of the installed package that the graph represents ~~and linking the new node to the existing node to indicate that the version corresponding to the new node depends on the version corresponding to the existing node~~;

subsequent to generating the graphs, determining that the graph corresponding to the first package of the operating system includes two paths that each include a node corresponding to the highest final version to which the first package may be updated;

~~subsequent to generating the graphs, utilizing each generated graph to identify a path that includes a node corresponding to the highest final version to which the package may be upgraded; and~~

subsequent to identifying the two paths, traversing each~~the~~ path and validating each node traversed;

upon validating each node of both paths, selecting the path that has the lowest cost of installation wherein each version in a path has a cost of installation and wherein the cost of installation of each path comprises the sum of the cost of installation of each version in the path; and

installing the update package corresponding to each node of the selected path to update the first package of the operating system. ~~such that:~~

~~upon each node of the path being validated, each version corresponding to the one or more nodes of the path are selected for installation; and~~

~~upon a node of the path failing validation, each node of the path is removed from the graph such that the versions corresponding to the removed nodes of the path are not selected for installation.~~

2. (Currently Amended) The method of claim 1 ~~wherein reviewing a collection of update packages that are ready to install further comprises~~ further comprising organizing the collection of update packages into coherent groups based on the currently installed package on the mobile computing device targeted by each update package.

3. (Canceled)

4. (Currently Amended) The method of claim 1 wherein ~~multiple paths that include a node corresponding to the highest final version exist and wherein utilizing each graph to identify a path comprises choosing the path having the fewest nodes~~ the path having the fewest nodes is selected.

5. (Canceled)

6. (Previously Presented) The method of claim 1 wherein validating each node comprises accessing a file associated with the update package corresponding to the node to determine whether the file is a suitable type of package file.

7. (Previously Presented) The method of claim 1 wherein validating each node comprises determining whether the update package corresponding to the node contains an associated manifest file that describes the update package contents.

8. (Canceled)

9. (Currently Amended) The method of claim 1, further comprising determining that none of the generated graphs correspond to the update package, and in response, generating a[[n]] new graph, and adding a node corresponding to the version of the update package to the new graph.

10. (Canceled)

11. (Previously Presented) The method of claim 1 wherein validating each node traversed comprises determining whether a certificate chain is associated with the given node via a parent node.

12. (Previously Presented) The method of claim 11 wherein a certificate chain is associated with the given node, and wherein validating each node traversed comprises verifying whether the package corresponding to the node contains a valid signature.

13. (Previously Presented) The method of claim 12 wherein the signature is valid for the given node, and further comprising, adding weighting data to the path based on that node.

14. (Previously Presented) The method of claim 13 further comprising, traversing the package graph to find a lowest weighted path, and adding information corresponding to that path to an update list.

15. (Previously Presented) The method of claim 11 wherein at least one node remains in the graph and wherein at least one remaining node is in a path having an associated weight value, and further comprising, traversing the graph to find a lowest weighted path, and adding information corresponding to that path to an update list.

16. (Previously Presented) The method of claim 1 wherein selecting each version for installation comprises determining a partition corresponding to each version, and sorting each version into lists by adding each version to a respective list based on the corresponding partition for that version.

17. (Previously Presented) One or more computer-readable storage media storing computer-executable instructions which when executed perform the method of claim 1.

18-19. (Canceled)

20. (Currently Amended) The ~~system~~ computer-readable storage media of claim ~~19~~ 17 wherein validating each node comprises accessing a file associated with the update package corresponding to the node to determine whether the file is a suitable type of package file.

21. (Currently Amended) The ~~system~~ computer-readable storage media of claim ~~19~~ 17 wherein validating each node comprises determining whether the update package corresponding to the node contains an associated manifest file that describes the update package contents.

22-27. (Canceled)

REMARKS

The Final Office Action, mailed January 2, 2009, 2008, considered claims 1-4, 6, 7, 9, 11-21 and 27. Claims 1-2, 4, 6, 7, 9, 11, 15-17 and 27 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Forbes et al. (US Patent Publication No.: 2002/0144248, hereinafter *Forbes*) in view of Te'eni (US Patent Publication No.: 2004/0015946, hereinafter *Te'eni*) in further view of Ottenstein et al. ("The Program Dependency Graph in a Software Development Environment", hereinafter *Ottenstein*). Claims 12-14 and 18-21 rejected under 35 U.S.C. § 103(a) as being unpatentable over Te'eni in view of Forbes in further view of O'Neill et al. (US Patent No.: 6,832,373, hereinafter *O'Neill*) and Ottenstein. Claim 3 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Te'eni in view of Forbes in futher view of Ottenstein and Weiss (Mark Allen Weiss, "Data Structures & Algorithm Analysis in C++", hereinafter *Weiss*).

By this response, claims 1, 2, 4, 20, and 21 are amended, while claims 3, 18, 19, and 27 are canceled. Claims 1, 2, 4, 6, 7, 9, 11-17, 20, and 21 remain pending of which claims 1 and 17 are independent.

Overview of the Invention and Discussion of Current Amendments

The present invention is directed to embodiments for updating the operating system of a mobile computing device (e.g. a Microsoft Windows CE .NET-based portable device, *see* pg. 9). The operating system is divided into discrete packages that may be individually updated. *See* pg. 10, lines 10-18. Multiple packages, including multiple versions of the same package, may be available at any time to update the operating system. Each update package generally is dependent on a lower version of the package whether that version is already installed on the device or is included as one of the update packages. In addition, there may exist separate paths (or groups) of update packages that may be installed that each update the package of the operating system to the same version.[1] The present invention is directed to a method for generating tree graphs which allow update paths to be determined, and for determining which path has a lower cost of installation. This is important due to the reduced memory and processing power of a typical mobile device.

[1] As was described in the previous response, multiple paths would exist when a version 9 of a package is available but that version may be reached by installing update packages from one of two paths (e.g. one path may be 6.0 – 6.1 – 6.5 – 7.0 – 8.0 – 9.0 and another may be 6.0 – 6.2 – 7.5 – 9.0).

The current amendments have been made to clarify that the operating system is divided into packages that are updated. In addition, the independent claims now are specifically directed to the case were multiple paths exist for updating a first installed package of the operating system. These amendments also better clarify the process of generating the tree graphs for each package of the operating system. For example, the language "recursively dependent on another update package" has been used to clarify that an update package can be placed in the tree graph at any distance from the base node as long as there are intermediate nodes from which the update package depends that lead to the base node. Also, the claims now explicitly state that each node corresponds to a different version of the package represented by the base node. Support for the amendments may be found on pages 22-32 or the specification which describe the processes depicted in figures 3-7.

Section 112 Rejections

The language that was rejected as being indefinite has been removed from each of the rejected claims. The claims now focus on the case where a single first package is updatable to a highest version using two different paths.

Prior Art Rejections

Although the cited art relates to the update of software, none of these references relates to determining how to update a package of an operating system to the highest possible version at the least cost. It is reemphasized that the current claims require that two paths, consisting of different versions to the same package, exist to reach the same highest version, and that the path requiring the least cost is selected such that only the update packages on that path are installed. None of the cited references teaches or suggests the determination of installation paths or the selection of the least costly path as claimed.

For example, in Forbes, a user selects which software he wants to be installed on the computer. When the software is selected (such as CoolestApp, *see* ¶¶ 43-44), it is determined whether dependencies exist for CoolestApp. If a dependency is detected, the user is notified such as by being directed to the website for the required software. The only portion of Forbes that is relevant to the generation of a graph is the listing of dependencies in the manifest file. *See* ¶ 84, Table 1. However, this listing is not used to generate a graph of the actual update packages, but is merely used to indicate that there are dependencies.

On the other hand, Te'eni does disclose the use of graphs to determine dependencies. These graphs are used to determine what components are required when a computer is updated. However, Te'eni does not use these graphs to determine a least costly path to update the computer. The distinction is best seem by comparing the objects of each invention. In the present invention, the goal is to determine, from multiple paths, which path is the least costly to update a single package of the operating system. In contrast, in both Forbes and Te'eni, the goal is to ensure that the user has all the required software in place so that different software selected for install will function properly. *See, e.g.* Te'eni, ¶ 11 ("there is an urgent need for an effective installation utility designed to resolve automatically *intercomponent* dependency conflicts....."). In other words, the present invention involves installing multiple versions of the same component, whereas the cited references involve determining whether different components are required for a component selected for installation to work. In both of the references, new components are being selected for install, not updated versions of the same component. For this reason, there would be no need for either Forbes or Te'eni to select the least costly path because there would only be a single path (e.g. CoolApp – CoolestApp).

In paragraph 48 of Te'eni, the sets of dependencies are described as lists of components that must be installed for a component to work. Although versions are mentioned here, the role of versions is completely different from the present invention. For example, the list may state that at least one editor must be installed from the set of vi, vim, or emacs, that vim may be either version 4 or 5, and that vim version 5 may be either 5.1, 5.2, or 5.3. In other words, the dependency list states that the component selected for install requires an editor that may be any of those listed, including any of the many version of vim. However, all that matters is that a single editor is installed. There is no need to determine how to get to the highest version of vim in the least cost because the check is being made to determine if a component other than vim will work properly on the system.

In summary, neither Forbes nor Te'eni relates to the determination of the optimal path for installing updates to a package of an installed operating system. In both references, dependencies between different components are checked. There is no generation of paths which include different versions of the same package. For these reasons, Forbes and Te'eni fail to teach or suggest each limitation of the independent claims.

The Ottenstein reference is similar to Forbes and Te'eni in that it relates to determining what data, files, or code, a program depends on. However, Ottenstein does not relate to generating graphs including multiple paths to the same version for updating an installed package of an operating system, or to the selection of the least costly path as claimed. The O'Neill and Weiss references were only cited to reject dependent claims and are also not relevant to these aspects of the independent claims.

In view of the foregoing, Applicant respectfully submits that all the rejections to the independent claims are now moot and that the independent claims are now allowable over the cited art, such that any of the remaining rejections and assertions made, particularly with respect to all of the dependent claims, do not need to be addressed individually at this time. It will be appreciated, however, that this should not be construed as Applicant acquiescing to any of the purported teachings or assertions made in the last action regarding the cited art or the pending application, including any official notice, and particularly with regard to the dependent claims.[2]

In the event that the Examiner finds remaining impediment to a prompt allowance of this application that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney at 801-533-9800.

The Commissioner is hereby authorized to charge payment of any of the following fees that may be applicable to this communication, or credit any overpayment, to Deposit Account No. 23-3178: (1) any filing fees required under 37 CFR § 1.16; and/or (2) any patent application and reexamination processing fees under 37 CFR § 1.17; and/or (3) any post issuance fees under 37 CFR § 1.20. In addition, if any additional extension of time is required, which has not otherwise been requested, please consider this a petition therefore and charge any additional fees that may be required to Deposit Account No. 23-3178.

Dated this 2nd day of April, 2009.

[2] Instead, Applicant reserves the right to challenge any of the purported teachings or assertions made in the last action at any appropriate time in the future, should the need arise. Furthermore, to the extent that the Examiner has relied on any Official Notice, explicitly or implicitly, Applicant specifically requests that the Examiner provide references supporting any official notice taken. Furthermore, although the prior art status of the cited art is not being challenged at this time, Applicant reserves the right to challenge the prior art status of the cited art at any appropriate time, should it arise. Accordingly, any arguments and amendments made herein should not be construed as acquiescing to any prior art status of the cited art.

Respectfully submitted,

RICK D. NYDEGGER
Registration No. 28,651
BRIAN D. TUCKER
Registration No. 61,550
Attorneys for Applicant
Customer No. 47973

RDN:BT:cj
2251804_1.DOC